# **README: Cloud & Shadow Segmentation using U-Net on Sentinel-2 Imagery**

## **📌 Project Overview**

This project involves building a U-Net-based deep learning model for segmenting clouds and cloud shadows in Sentinel-2 satellite images using a filtered subset of the CloudSEN12 dataset. The dataset has been processed and loaded using the TacoReader format, and training is performed using 9 Sentinel-2 bands. The model performs multi-class segmentation, predicting one of three classes for each pixel:

* **0:** Clear
* **1:** Cloud/Thin Cloud
* **2:** Shadow

The goal is to generate accurate cloud masks that can support downstream applications like agricultural monitoring, environmental analysis, or cloud-free mosaicking.

While the full CloudSEN12 dataset is large and diverse, we focused on a smaller subset limited to **Mexico**. This choice allowed for efficient experimentation and iteration. Despite the restricted geographic scope, the model still performed well, producing meaningful and visually accurate cloud and shadow masks across validation and test samples.

## **⚙️ Instructions to Run Code**

**Install dependencies** pip install torch torchvision rasterio matplotlib scikit-learn geopandas tacoreader

1. **Prepare dataset**
   * Use TacoReader to filter and export a subset (mini.taco) of CloudSEN12 with cloud/shadow presence over a specific region.

**Run the training script** python train\_unet.py

1. This will:  
   * Train the U-Net model on 9 Sentinel-2 bands
   * Save best model weights in checkpoints/best\_model.pth
   * Save final model in final\_model\_weights.pth
   * Log sample predictions and metrics
2. **Evaluate the model** After training, the best model is loaded and evaluated on a held-out test set.

## **🧪 Data Preprocessing Steps**

* Loaded .taco dataset using tacoreader.
* Selected 9 bands: [B2, B3, B4, B5, B6, B7, B8, B11, B12]
* Normalized EO data to [0, 1] by dividing by 10000 and clipping.
* Converted raw 8-class labels into 3-class segmentation:  
  + 0 → 0 (clear)
  + [1, 2] → 1 (cloud/thin cloud)
  + 3 → 2 (shadow)
* Skipped samples with no valid labels (i.e., all 255)

## **🧠 Model Used**

### **U-Net**

* A standard U-Net architecture with:  
  + 4 encoder/decoder layers
  + BatchNorm and ReLU activations
  + 9 input channels and 3 output classes
* Loss: CrossEntropyLoss (ignoring class 255)
* Optimizer: AdamW
* Metrics: Pixel Accuracy, F1 Score (macro), IoU (macro)

## **📊 Key Findings or Observations ✅**

* Using 9 Sentinel-2 bands yielded significantly better results than using only RGB (3-band) or RGB+NIR (4-band) inputs. The inclusion of short-wave infrared (SWIR) and red-edge bands provided better spectral discrimination for detecting thin clouds and shadows, especially in complex scenes.
* Loss NaN issues were resolved with input normalization and ignore\_index=255.
* Validation IoU was a reliable indicator for model checkpointing.
* Some test set samples remain challenging due to label sparsity or overclouding.

## 

## 

## **🧾 Output Results**

### **Final Validation Set Metrics (Epoch 35/40):**

* **Accuracy:** 0.888
* **F1 Score (macro):** 0.837
* **IoU (macro):** 0.754

### **Final Test Set Metrics (best checkpoint loaded):**

* **Accuracy:** 0.8606
* **F1 Score (macro):** 0.7423
* **IoU (macro):** 0.6527

### **Sample Predictions**

* 20 sample predictions were saved under sample\_outputs/ with:  
  + Input RGB patch: input\_<idx>.png
  + Ground truth label: gt\_<idx>.png
  + Model prediction: pred\_<idx>.png

All outputs are visualized using RGB bands [B4, B3, B2] for better interpretability.

## 

## 

## 

## **🛠️ Tools & Frameworks Used**

* **PyTorch** – for model building and training
* **Torchvision** – for saving images
* **TacoReader** – to handle .taco format and STAC metadata
* **Rasterio** – to load EO and label rasters
* **Matplotlib** – to visualize RGB, GT, and predictions
* **scikit-learn** – for computing F1 and IoU metrics